TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

A Denotational Model for Mobile Point-to-Point Data-flow Networks with Channel Sharing

Radu Grosu, Ketil Stølen, Manfred Broy

TUM-I9724
SFB-Bericht Nr. 342/17/97 A
Mai 97

TUM–INFO–05-I9724-350/1.–FI

Alle Rechte vorbehalten
Nachdruck auch auszugsweise verboten

©1997 SFB 342 Methoden und Werkzeuge für
die Nutzung paralleler Architekturen

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

A Denotational Model for Mobile Point-to-Point Data-flow Networks with Channel Sharing

Radu Grosu, Ketil Stølen, Manfred Broy

Institut für Informatik, TU München, D-80290 München
email:grosu,stoelen,broy@informatik.tu-muenchen.de

Abstract

We present a fully abstract, denotational model for mobile, timed, nondeterministic data-flow networks whose components communicate in a point-to-point fashion. In this model components and networks of components are represented by sets of stream processing functions. Each stream processing function is required to be strongly pulse-driven and privacy preserving. A function is strongly pulse-driven if it is contractive with respect to the metric on streams. This property guarantees the existence of unique fix-points. The privacy preservation property can be thought of as an invariant specific to mobile point-to-point systems. Firstly, it guarantees that a function never accesses, depends on or forwards a port whose name it does not already know. Secondly, it guarantees that at the same point in time no port is known to more than two components, namely the sender and the receiver. Our model allows the description of a wide variety of networks — in particular, the description of unbounded nondeterministic networks. We demonstrate some features of our model by specifying a communication central.

1 Introduction

One of the most prominent theories for interactive computation is the theory of data-flow networks. In this theory, an interactive system is represented by a network of autonomous components which communicate solely by asynchronous transmission of messages via directed channels.

A very elegant model for static, deterministic data-flow networks, whose components communicate in a point-to-point fashion, was given by Kahn in [Kah74]. Despite of its elegant foundation, this class of networks is, however, too restrictive for many practical applications. In this paper we extend Kahn's model in a number of ways.

Firstly, contrary to Kahn, we model nondeterministic behavior. Like Park [Par83], Broy [Bro87] and Russell [Rus90], we represent nondeterministic data-flow networks by sets

of stream processing functions. However, in contrast with [Par83] and Broy [Bro87], our model is fully abstract. This is achieved, by considering only sets of functions which are closed with respect to the external observations. The closure idea was used by Russell for the same purpose. However, contrary to Russell, we use a timed model and a different notion of observation. This allows us to describe a considerably greater class of networks. In particular, we can describe unbounded nondeterministic networks. Moreover, since our model is fully abstract, we obviously avoid the expressiveness problem known as the Brock/Ackermann anomaly [BA81].

Secondly, contrary to Kahn, we describe dynamically reconfigurable or mobile networks — networks in which every component may change its communication partners on the basis of computation and interaction. The formal modeling of mobility has been a very popular research direction in recent years. However, most models published so far have been formalized mainly in operational terms. Examples of such models are the Actor Model [HBS73, AMST92], the π–Calculus [EN86, MPW92a, MPW92b], the Chemical Abstract Machine [BB90], the Rewriting Logic [Mes91] and the Higher Order CCS [Tho89]. On the contrary, our model gives a denotational formalization of mobility. As in the above models, this formalization is based on two assumptions. Firstly, ports are allowed to be passed between network components. Secondly, the components preserve privacy: their behaviors do not depend on ports they do not know. Although it is well understood how to express privacy operationally, there is less denotational understanding. Informally speaking, our solution is to require each stream processing function to never receive on, send via or forward a port whose name "it does not already know". By "the ports it does not already know" we basically mean any port which is not in its initial interface, it has not already received and it has not already created itself. Any port created by the function itself is assigned a "new" name taken from a set "private" to the component in question. This can be thought of as a privacy invariant satisfied by any mobile system.

In a companion paper [GS95a] we have shown how many-to-many communication can be modeled in this setting. In this paper we concentrate on point-to-point communication. There are basically two different variants of point-to-point communication. In the first case, the sender and the receiver of a channel remain the same during the whole lifetime of the channel. In the second case, the sender and the receiver of a channel may change. However, at any point in time a channel has not more than one sender and one receiver. In the first case there is no interference at all — two different components cannot send along the same channel. In the second case only a restricted type of interference may occur — two different components may send on the same channel, but never simultaneously. The advantage of the first alternative is its simplicity with respect to formal reasoning and understanding. The advantage of the second alternative is that many things can be expressed more directly. In this paper we concentrate on the second alternative. The first alternative is investigated in [GS95b].

Point-to-point communication is guaranteed by imposing local constraints on the way

ports are communicated. Some readers may wonder why we at all find point-to-point communication interesting. After all, point-to-point communication is only a special case of many-to-many communication. The main reason is that point-to-point communication allows a tight control of channel interference. In a point-to-point model the default situation is no interference at all (in the same time unit). Interference is only introduced via explicit fair merge components for those channels where this is desirable. In a many-to-many model there is interference by default. The tight control of interference in a point-to-point setting simplifies both specification (programming) and formal reasoning. Thus, our interest in point-to-point communication is methodological: we want to combine the power of nondeterminism and mobility with the simplicity of point-to-point reasoning.

Although we could have formulated our semantics in a cpo context, we decided to base it on the topological tradition of metric spaces [Niv82, dBZ82, AdBKR89]. Firstly, we wanted to understand the exact relationship between our approach and those based on metric spaces. Secondly, the use of metric spaces seems more natural since our approach is based on infinite streams, and since our pulse-drivenness constraint, guaranteeing the existence of a unique fix-point, corresponds straightforwardly to contractivity.

2 Basic Notions

We model an interactive system by a network of autonomous components which communicate via directed channels in a time-synchronous and message-asynchronous way. Time-synchrony is achieved by using a global clock splitting the time axis into discrete, equidistant time units. Message-asynchrony is achieved by allowing arbitrary, but finitely many messages to be sent along a channel in each time unit.

2.1 Communication Histories

We model the communication histories of directed channels by infinite streams of finite streams of messages. Each finite stream represents the communication history within a time unit. The first finite stream contains the messages received within the first time unit, the second the messages received within the second time unit, and so on. Since time never halts, any complete communication history is infinite.

Let M be the set of all messages. Then $[M^*]$ is the set of all complete communication histories and $(M^*)^*$ is the set of all partial communication histories[1].

In the introduction we anticipated that components are allowed to communicate *ports*. A port is a *channel name* together with an *access right*, which is either a read right,

[1]For an arbitrary set S, by S^* we denote the set of finite streams over S, by $[S]$ we denote the set of infinite streams over S, and by S^ω we denote $S^* \cup [S]$. See also the appendix.

represented by ?, or a write right, represented by !. Let N be the set of all channel names, then $?N = \{?n \mid n \in N\}$ is the corresponding set of read ports and $!N = \{!n \mid n \in N\}$ is the corresponding set of write ports. We also write $?!N$ for $?N \cup !N$. We assume that $?!N \subseteq M$. Let D be the set of all messages not contained in the set of ports, i.e. $D = M \setminus ?!N$.

Since ports are exchanged dynamically between network components, each component can potentially access any channel in N. For that reason we model the complete input and output histories of a component by named stream tuples contained in $N \to [M^*]$. The partial ones are modeled by $N \to (M^*)^*$. In the sequel we will refer to named stream tuples of these signatures as named communication histories. Thus each named communication history assigns a communication history to each channel name in N.

2.2 Pulse-Driven Functions

A *deterministic component* is modeled by a stream processing function

$$f \in (N \to [M^*]) \to (N \to [M^*])$$

mapping complete named communication histories for its input channels to complete named communication histories for its output channels.

In accordance with the operational intuition, the functions process their input *incrementally* — at any point in the time, their output is not allowed to depend on future input. Functions satisfying this constraint are called *weakly pulse-driven.* If the output they produce in time unit t, is not only independent of future input, i.e. the input received at time $t+1$ or later, but also of the input received during time unit t, then they are called *strongly pulse-driven.*

Definition 1 (Pulse-driven functions) *A function $f \in (N \to [M^*]) \to (N \to [M^*])$ is weakly pulse-driven if*

$$\forall \theta, \varphi, j : \ \theta{\downarrow_j} = \varphi{\downarrow_j} \quad \Rightarrow \quad f(\theta){\downarrow_j} = f(\varphi){\downarrow_j},$$

and strongly pulse-driven if

$$\forall \theta, \varphi, j : \ \theta{\downarrow_j} = \varphi{\downarrow_j} \quad \Rightarrow \quad f(\theta){\downarrow_{j+1}} = f(\varphi){\downarrow_{j+1}},$$

where $\theta{\downarrow_j}$ represent the prefix of θ of length j. □

We use the arrow $\multimap$ for the set of strongly pulse-driven functions.

Theorem 1 *A stream processing function is strongly pulse-driven iff it is contractive with respect to the metric of streams. A stream processing function is weakly pulse-driven iff it is non-expansive with respect to the metric of streams.*

Proof sketch: Straightforward. The metric of named stream-tuples is defined in the appendix. □

2.3 Sum

The sum operator takes two named communication histories as input and delivers their "union" as output. We define both a partial "disjoint" sum and a total sum.

Definition 2 (Partial sum) *Given two named stream tuples* $\varphi, \psi \in (N \to [M^*])$ *such that*

$$\forall i, n : \ \varphi(i)(n) = \epsilon \ \vee \ \psi(i)(n) = \epsilon.$$

We define their partial sum $\varphi + \psi$ *to denote the element of* $N \to [M^*]$ *such that for all* $i \in N, n \in \mathbb{N}$*:*

$$(\varphi + \psi)(i)(n) = \begin{cases} \varphi(i)(n) & \text{if } \varphi(i)(n) \neq \epsilon \\ \psi(i)(n) & \text{if } \varphi(i)(n) = \epsilon \end{cases}$$ □

Note that the partial sum has no syntactic conditions assuring its well-definedness. We therefore define a total version $\varphi \Leftrightarrow \psi$. This simplifies the use of the Banach's fixed point theorem. Totalisation is achieved by defining $(\varphi \Leftrightarrow \psi)(i)(n)$ to be ϵ if both $\varphi(i)(n)$ and $\psi(i)(n)$ are different from ϵ.

Definition 3 (Total sum) *Given two named stream tuples* $\varphi, \psi \in (N \to [M^*])$*. We define the* total sum $\varphi \Leftrightarrow \psi$ *to denote the element of* $N \to [M^*]$ *such that for all* $i \in N, n \in \mathbb{N}$*:*

$$(\varphi \Leftrightarrow \psi)(i)(n) = \begin{cases} \psi(i)(n) & \text{if } \varphi(i)(n) = \epsilon \\ \varphi(i)(n) & \text{if } \varphi(i)(n) \neq \epsilon \wedge \psi(i)(n) = \epsilon \\ \epsilon & \text{if } \varphi(i)(n) \neq \epsilon \wedge \psi(i)(n) \neq \epsilon \end{cases}$$ □

Note that $\varphi \Leftrightarrow \psi$ has a hiding effect if there are i and n such that $\varphi(i)(n) \neq \epsilon \wedge \psi(i)(n) \neq \epsilon$, and that $\varphi \Leftrightarrow \psi$ is equal to $\varphi + \psi$, otherwise.

Theorem 2 *The total sum operation is weakly pulse-driven.*
Proof: The sum $(\varphi \Leftrightarrow \psi)(i)(n)$ depends only on $\varphi(i)(n)$ and $\psi(i)(n)$. □

3 Privacy Preservation

A stream processing function $f \in (N \to [M^*]) \to (N \to [M^*])$ used to model a component is not only required to be strongly pulse-driven, but also to be privacy preserving. Firstly, privacy preservation guarantees that it accesses, depends on and forwards only ports it already knows. Thus, privacy preservation characterizes the way a function gains access to a port. Secondly, privacy preservation makes sure that a port is forgotten as soon as it is sent. This ensures the point-to-point invariant, namely that at any point in time the same port is known to exactly one function. Thus, privacy preservation also characterizes the way port access is lost.

3.1 Gaining Port Access

The way port access is gained can be described with respect to Figure 1, as follows. Initially, each stream processing function reads from a designated set of input channels I and writes on a designated set of output channels O. These two sets name the *static* channels or the initial wiring. We require that $I \cap O = \emptyset$. To make sure that channels *created* by the different components in a network have different names, each stream processing function is assigned a set of *private names* P. Obviously, this set should be disjoint from the static interface. Thus we require that $(I \cup O) \cap P = \emptyset$.

During the computation, the sets of accessible channels can grow. For example, if the function receives a read port $?i$ then it may read from the channel i, and if it receives a write port $!o$ then it may write on the channel o. Similarly, whenever the function sends an output port $!j$, whose channel $j \in P$ it has created itself, it may later read what is sent along j, or whenever it sends an input port $?p$, whose channel $p \in P$ it has created itself, it may itself send messages along p which eventually are read by the component which receives the input port.

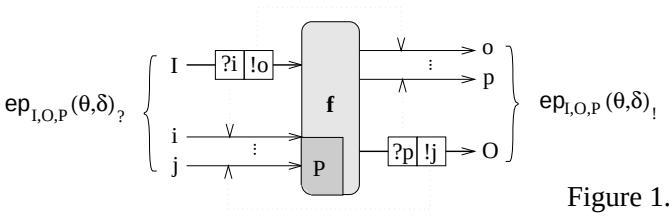

Figure 1.

At any given point in time n and named input history θ, by $\mathsf{ep}_{I,O,P}(\theta, f(\theta))(n)$ we denote the set of accessible external ports of f and by $\mathsf{pp}_{I,O,P}(\theta, f(\theta))(n)$ we denote the set of private ports of f. When ambiguities do not occur we often refer to these sets as ep and pp, respectively. For any set of ports S, $S_?$ and $S_!$ denote the subsets of input and output ports, respectively.

3.2 Loosing Port Access

To communicate in a *point-to-point* fashion a network of stream processing functions has to maintain the following invariant: at any given point in time, each port is "known" to at most one function. This means that for any channel c, at any point in time only two functions may access c, namely the sender and the receiver.

This point-to-point invariant is ensured by imposing local requirements on the behavior of functions.

Suppose f sends one of its external ports $p \in \mathsf{ep}$ to some other function g (see Figure 2). If $p = !o$ then both f and g have send access to o. Thus, f may interfere with g which is not what we want. Interference can also be the result if $p = ?i$ because in that case

both f and g may at some point in the future receive the same write port $!o$ on i. To avoid this risk of interference without loosing compositionality we have to restrict any function to "forget" any port it sends along its output channels. Thus, with respect to our example, as soon as f forwards p, it can no longer take advantage of this port, i.e., p is deleted from ep.

Note that a function may send the same port several times because it may gain access to the same port several times. However, it may not send it more than once for each time it gains access to it. For example, if a function f initially has access to a port p and f forwards this port, then f must postpone sending it again until it has regained access to p by receiving p on one of its input ports.

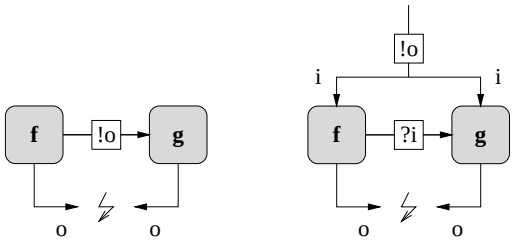

Figure 2.

Private ports may become public and public ports may become private. A function f may make a port $p \in$ pp public by sending it to some other function g. In that case both p and $\widetilde{p}$[2] are deleted from pp, $\widetilde{p}$ is included in ep and p is "forgotten".

A public port becomes private if some function f receives some port p whose complement $\widetilde{p}$ it already has in its set of public ports. After all, if f has both ports to a channel, then only f knows about this channel. Consequently, both p and $\widetilde{p}$ should be added to its set of private ports pp and $\widetilde{p}$ should be deleted from its set of public ports ep.

We have described how functions gain access to and forget ports, and also how public ports may become private and the other way around. These are all local constraints. Since the function runs in an open environment these constraints are of course not sufficient unless the environment also plays the point-to-point game. After all, our local constraints are of no help if the environment behaves arbitrary. One way to deal with this problem is to impose an environment assumption and only require privacy preservation when this assumption is fulfilled. However, since we are only interested in environments which stick to the rules, we constrain our functions to ignore the input messages which do not respect the privacy restrictions.

We now explain how these constraints are imposed formally. First of all, since a function can only send the same port once for each time it gains access to a port, we only have to consider named communication histories in which the same port does not occur twice in the same time unit. A named communication history $\theta \in N \to [M^*]$ in which the same port does not occur twice in the same time unit, i.e.,

[2]Given a set of ports $S \subseteq ?!N$. Then $\overline{S} = N \setminus S$ and $\widetilde{S} = \{\widetilde{p} \mid p \in S\}$ where $\widetilde{!i} = ?i$ and $\widetilde{?i} = !i$. We also regard $\theta(i)(n)$, when conveniently, as a set.

$$\forall n, p, i, j : \; p \in \theta(i)(n) \wedge p \in \theta(j)(n) \quad \Rightarrow \quad i = j$$

is said to be *port-unique*. We use the arrow $\stackrel{\mathsf{u}}{\rightarrow}$ to distinguish named communication histories satisfying this port uniqueness constraint.

Port-unicity is preserved by the total sum if the two arguments never have an occurrence of the same port within the same time unit. More precisely, let the operator prt collect the ports occurring in a named communication history for each time n

$$\mathsf{prt}(\theta)(n) = \{p \; \in \; ?!N \; \mid \; \exists i : \; p \in \theta(i)(n)\}.$$

Then for $\varphi, \psi \in N \stackrel{\mathsf{u}}{\rightarrow} [M^*]$ such that for all n, $\mathsf{prt}(\theta)(n) \cap \mathsf{prt}(\phi)(n) = \emptyset$, it is the case that $\varphi \leftrightarrows \psi \in N \stackrel{\mathsf{u}}{\rightarrow} [M^*]$.

Definition 4 (External and private ports) *Given I, O, P, θ and δ. We define*

$$\mathsf{ep}_{I,O,P}(\theta,\delta)(n) = \mathsf{ep}_n, \quad \mathsf{pp}_{I,O,P}(\theta,\delta)(n) = \mathsf{pp}_n$$

where ep_n and pp_n are given below.

$$\begin{array}{lcllcl}
\mathsf{ep}_1 & = & ?I \; \cup \; !O, & \mathsf{pp}_1 & = & ?!P, \\
\mathsf{ep}_{n+1} & = & (\mathsf{ep}_n \cup r_n \cup g_n) \setminus (s_n \cup h_n), & \mathsf{pp}_{n+1} & = & (\mathsf{pp}_n \cup h_n) \setminus (s_n \cup \widetilde{s_n}),
\end{array}$$

where

$$r_n = \bigcup_{?i \in \mathsf{ep}_n} \{p \mid p \in \overline{\mathsf{pp}_n \cup \mathsf{ep}_n} \cap \theta(i)(n)\}, \quad h_n = \{p, \widetilde{p} \mid p \in r_n \wedge \widetilde{p} \in \mathsf{ep}_n\},$$
$$s_n = \bigcup_{!i \in \mathsf{ep}_n} \{p \mid p \in (\mathsf{pp}_n \cup \mathsf{ep}_n) \cap \delta(i)(n)\}, \quad g_n = \{\widetilde{p} \mid p \in s_n \wedge p \in \mathsf{pp}_n\}$$

The sets r_n, s_n, g_n and h_n are the sets of received, sent, generated and to-be-hidden ports. □

Let $\mathsf{dom}_{I,O,P}(\theta, f(\theta))$ be the named input communication history actually read by f and $\mathsf{rng}_{I,O,P}(\theta, f(\theta))$ be the named output communication history which should be produced by f. The restricting functions dom and rng which are imposed to θ and $f(\theta)$ are defined below.

Definition 5 (Domain and range) *Given I, O, P, θ and δ. Let $\mathsf{ep}_n = \mathsf{ep}_{I,O,P}(\theta,\delta)(n)$ and $\mathsf{pp}_n = \mathsf{ep}_{I,O,P}(\theta,\delta)(n)$. Then*

$$\mathsf{dom}_{I,O,P}(\theta,\delta)(i)(n) = \begin{cases} (\overline{\mathsf{pp}_n \cup \mathsf{ep}_n} \cup D) \; \textcircled{c} \; \theta(i)(n) & \text{if} \quad ?i \in \mathsf{ep}_n \\ \epsilon & \text{otherwise} \end{cases}$$

$$\mathsf{rng}_{I,O,P}(\theta,\delta)(i)(n) = \begin{cases} (\mathsf{pp}_n \cup \mathsf{ep}_n \cup D) \; \textcircled{c} \; \delta(i)(n) & \text{if} \quad !i \in \mathsf{ep}_n \\ \epsilon & \text{otherwise} \end{cases}$$

□

Theorem 3 *The functions* dom *and* rng *are weakly pulse-driven.*

Proof sketch: $\mathsf{dom}_{I,O,P}(\theta,\delta)(n)$, $\mathsf{rng}_{I,O,P}(\theta,\delta)(n)$ depend only on $\theta{\downarrow_n}$ and $\delta{\downarrow_n}$. □

Theorem 4 *The functions* dom *and* rng *have the following properties:*

$$\begin{aligned}\mathsf{dom}_{I,O,P}(\theta,\delta) &= \mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta,\delta),\delta) = \mathsf{dom}_{I,O,P}(\theta,\mathsf{rng}_{I,O,P}(\theta,\delta)),\\ \mathsf{rng}_{I,O,P}(\theta,\delta) &= \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta,\delta),\delta) = \mathsf{rng}_{I,O,P}(\theta,\mathsf{rng}_{I,O,P}(\theta,\delta)).\end{aligned}$$

Proof sketch: By induction on the recursive definition of $\mathsf{dom}_{I,O,P}$ and $\mathsf{rng}_{I,O,P}$. □

Definition 6 (Privacy preserving) *A function* $f \in (N \to [M^*]) \to (N \to [M^*])$ *is called* privacy preserving *with respect to the initial wiring* (I,O) *and the private names* P *iff:*

$$\forall\theta : f(\theta) = f(\mathsf{dom}_{I,O,P}(\theta, f(\theta))) = \mathsf{rng}_{I,O,P}(\theta, f(\theta)).$$ □

Definition 7 (Mobile functions) *A function* $f \in (N \xrightarrow{\mathsf{u}} [N^*]) \to (N \xrightarrow{\mathsf{u}} [N^*])$ *is said to be* mobile *with respect to* (I,O,P)*, where* $I \cap O = P \cap (I \cup O) = \emptyset$*, if it is strongly pulse-driven and privacy preserving. We use the arrow* $\overset{I,O,P}{\to}$ *to distinguish functions that are mobile with respect to* (I,O,P) *from other functions.* □

4 Mobile Components

We model a nondeterministic component by a set of mobile functions F. Any pair $(\theta, f(\theta))$, where $f \in F$, is a possible *behavior* of the component. Intuitively, for each input history each mobile function $f \in F$ represents one possible nondeterministic behavior.

Different sets of mobile functions may have the same set of behaviors. The reason is that for some sets of mobile functions we may find additional mobile functions which can be understood as combinations of the functions already in the set. For example, we may find a mobile function g which for one input history behaves as the function $f \in F$ and for an other input history behaves as the function $f' \in F$, and so on. This means, a model in which a nondeterministic component is represented by an arbitrary set of mobile functions, is too distinguishing, and consequently, not fully abstract. To achieve full abstraction we consider only closed sets, i.e. sets F, where each combination of functions in F, which gives a mobile function, is also in F.

Definition 8 (Mobile components) *A mobile component, with initial wiring* (I,O) *and private names* P*, where* $P \cap (I \cup O) = \emptyset$*, is modeled by a nonempty set of mobile functions*

$$F \subseteq (N \xrightarrow{\mathsf{u}} [M^*]) \overset{I,O,P}{\to} (N \xrightarrow{\mathsf{u}} [M^*])$$

that is closed *in the sense that for any mobile function* f *of the same signature*

$$(\forall\theta \in (N \to [M^*]) : \exists f' \in F : f(\theta) = f'(\theta)) \Rightarrow f \in F.$$ □

5 Composition

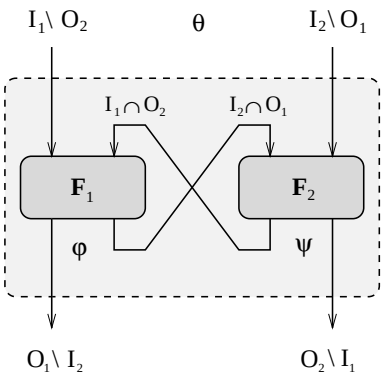

Figure 3.

Definition 9 (Point-to-point composition) *Given two mobile components:*

$$F_1 \subseteq (N \xrightarrow{\mathsf{u}} [M^*])^{\,I_1,O_1,P_1} \overset{}{\rightarrow} (N \xrightarrow{\mathsf{u}} [M^*]), \qquad F_2 \subseteq (N \xrightarrow{\mathsf{u}} [M^*])^{\,I_2,O_2,P_2} \overset{}{\rightarrow} (N \xrightarrow{\mathsf{u}} [M^*])$$

such that $I_1 \cap O_1 = I_2 \cap O_2 = I_1 \cap I_2 = O_1 \cap O_2 = \emptyset$,
$P_1 \cap (I_2 \cup O_2 \cup P_2) = P_2 \cap (I_1 \cup O_1 \cup P_1) = \emptyset$. *The* composition *of* F_1 *and* F_2 *gives a network whose static structure is characterized by Figure 3. It is formally defined as follows:*

$$\begin{array}{lll} I & = (I_1 \setminus O_2) \cup (I_2 \setminus O_1), & O \;\; = (O_1 \setminus I_2) \cup (O_2 \setminus I_1), \\ IO & = (I_1 \cap O_2) \cup (I_2 \cap O_1), & P \;\; = P_1 \cup P_2 \cup IO \end{array}$$

$$\begin{array}{l} F_1 \otimes F_2 = \{f \in (N \xrightarrow{\mathsf{u}} [M^*])^{\,I,O,P} \overset{}{\rightarrow} (N \xrightarrow{\mathsf{u}} [M^*]) \;\mid\; \forall \theta : \exists f_1 \in F_1, f_2 \in F_2 : \\ \qquad f(\theta) = \mathsf{rng}_{I,O,P}(\theta, \varphi + \psi) \;\; \mathsf{where} \\ \qquad\quad \varphi = f_1(\vartheta + \psi), \quad \psi = f_2(\vartheta + \varphi), \quad \vartheta = \mathsf{dom}_{I,O,P}(\theta, \varphi + \psi)\} \end{array}$$ □

Note the role of $\mathsf{dom}_{I,O,P}$ and $\mathsf{rng}_{I,O,P}$ in maintaining privacy. If F_1 sends a private port $!o$ on a feedback channel, then only F_2 should send along o and only F_1 should receive on o. F_1 can receive on o because $\mathsf{dom}_{I_1,O_1,P_1}$ is automatically enlarged with o. Only F_1 can receive on o because $\mathsf{rng}_{I,O,P}$ automatically hides what F_2 sends along o. F_2 can send along o because $\mathsf{rng}_{I_2,O_2,P_2}$ is automatically enlarged with o. Only F_2 can influence F_1 on o because $\mathsf{dom}_{I,O,P}$ automatically hides what the environment sends along o.

Similarly, if F_1 sends a private port $?i$ on a feedback channel, then only F_2 should receive on i and only F_1 should send along i. F_2 can receive on i because $\mathsf{dom}_{I_2,O_2,P_2}$ is automatically enlarged with i. Only F_2 can receive on i because $\mathsf{rng}_{I,O,P}$ automatically hides what F_1 sends along i. F_1 can send along i because $\mathsf{rng}_{I_1,O_1,P_1}$ is automatically enlarged with i. Only F_1 can influence F_2 on i because $\mathsf{dom}_{I,O,P}$ automatically hides what the environment sends along i.

Theorem 5 *$F_1 \otimes F_2$ is a mobile component.*
Proof: See appendix. □

6 Communication Central

As an example we specify a communication central (see Figure 4). Its task is to build up connections between $\mathsf{station}_1$ to $\mathsf{station}_n$. Each $\mathsf{station}_k$ is connected to the central with an output channel o_k and an input channel i_k. These are the initial "wires". Along their output channel o, the stations can send ports to be connected (both read and write) to the central which according to an internal prophecy distributes them to the other stations.

Let $!c$ be a write port sent by $\mathsf{station}_1$ to the central, which forwards it to $\mathsf{station}_2$. Then $\mathsf{station}_2$ can send along the channel c and $\mathsf{station}_1$ can receive on the channel c. Hence, these two stations are dynamically connected.

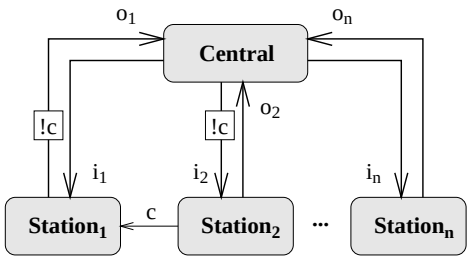

Figure 4.

In order to specify the central, let us introduce two basic operators. The first operator is a *filtration operator* for tuples of streams. For any set of n-tuples of messages A and n-tuple of streams s by $A©s$ we denote the result of truncating each stream in s at the length of the shortest stream in s, and selecting or deleting s_j depending on whether s_j is in A or not. By s_j we denote the tuple of messages whose kth component is equal to the jth message of the kth component stream of s. For example,

$$\{(a,a)\}©(\langle a,b,a,b,a,b,a\rangle,\langle a,a,a\rangle) = (\langle a,a\rangle,\langle a,a\rangle).$$

The second operator is a *time abstraction operator*: for any named communication history β, $\overline{\beta}$ denotes the result of removing all time information in β. For any i, this is achieved by concatenating all the finite streams in $\beta(i)$ into one stream[3]. Thus each communication history consisting of infinitely many finite streams of messages is replaced by the result of concatenating its finite streams into one stream of messages. As a result the timing information is abstracted away.

$$\text{central} \stackrel{\text{def}}{=} \{f \in (N \stackrel{\cup}{\to} [M^*]) \stackrel{\{o_1,\ldots,o_n\},\{i_1,\ldots,i_n\},\emptyset}{\rightharpoonup} (N \stackrel{\cup}{\to} [M^*]) \mid \forall\theta : R(\overline{\theta}, \overline{f(\theta)})\},$$

where

$$\begin{aligned} R(\alpha,\beta) \stackrel{\text{def}}{=}\ & \exists p_in, p_out \in [\{1,\ldots,n\}], \exists int_buf \in M^\omega : \forall j \in [1..n] : \\ & \alpha(o_j) = (M \times \{j\})©(int_buf, p_in) \wedge \\ & \beta(i_j) = (M \times \{j\})©(int_buf, p_out). \end{aligned}$$

Note that this specification does not say anything about the timing of the output. The existentially quantified variable int_buf can be understood as an internal buffer in which

[3]Do not confuse the time abstraction operator with set difference.

the input messages are placed in accordance with the oracle *p_in*. The other oracle *p_out* characterizes the way these messages are distributed to the stations.

7 Discussion

We have had several sources of inspiration. First of all, the modeling of nondeterministic networks is inspired by the semantic models for static nondeterministic networks [Par83], [Kok87], [Bro87]. Park models components by sets of functions in the same way as we do. However, he models time with time ticks $\surd$ and his functions are defined also for finite streams. Moreover, infinite streams are not required to have infinitely many ticks. Kok models nondeterministic components by functions mapping communication histories to sets of communication histories. We use instead a closed set of deterministic pulse-driven functions. This allows us to model unbounded nondeterminism without having to introduce a more complex metric. [Bro87] employs sets of functions as we do, but these functions work on untimed finite and infinite streams. This makes the model more abstract, but at the same time more complex with respect to its theoretical basis. The formulation of pulse-drivenness has been taken from [Bro95a], and the use of named communication histories is based on [BD92].

Our ideas on mobility have also had several sources. [Bro95b] and [Gro94] give an equational characterization of dynamic reconfiguration. [Gro94] also presents a semantic model for mobile, deterministic networks. However, that model is higher-order and mobility is achieved by communicating channels and functions instead of ports.

The idea of communicating names (ports) was inspired by [MPW92a, MPW92b]. The action structures [Mil92] are also related to our model. The idea of associating an access right with each channel in order to control interference was taken from [SKL90]. The semantic model of [SRP91] has also some similarities to our model. However, the intentions are complementary. [SRP91] aims at generality. We, on the other hand, have developed a particular model based on traditional stream processing functions. In [SRP91] inconsistencies can only be avoided by syntactic constraints. In our model inconsistencies cannot occur.

References

[AdBKR89] P. America, J. de Bakker, J. N. Kok, and J. Rutten. Denotational semantics of a parallel object-oriented language. *Information and Computation*, 83:152–205, 1989.

[AMST92] G. Agha, I. A. Mason, S. F. Smith, and C. L. Talcott. Towards a theory of actor computation. In *Proc. CONCUR'92, Lecture Notes in Computer Science 630*, pages 565–579, 1992.

[BA81] J. D. Brock and W. B. Ackermann. Scenarios: A model of non-determinate computation. In *Proc. Formalization of Programming Concepts, Lecture Notes in Computer Science 107*, pages 252–259, 1981.

[BB90] G. Berry and G. Boudol. The chemical abstract machine. In *Proc. POPL'90*, pages 81–94, 1990.

[BD92] M. Broy and C. Dendorfer. Modelling operating system structures by timed stream processing functions. *Journal of Functional Programming*, 2:1–21, 1992.

[Bro87] M. Broy. Semantics of finite and infinite networks of concurrent communicating agents. *Distributed Computing*, 2:13–31, 1987.

[Bro95a] M. Broy. Advanced component interface specification. In *Proc. TPPP'94, Lecture Notes in Computer Science 907*, pages 369–392, 1995.

[Bro95b] M. Broy. Equations for describing dynamic nets of communicating systems. In *Proc. 5th COMPASS Workshop, Lecture Notes in Computer Science 906*, pages 170–187, 1995.

[dBZ82] J. W. de Bakker and J. I. Zucker. Denotational semantics of concurrency. In *Proc. 14 ACM Symposium on Theory of Computing*, pages 153–158, 1982.

[EN86] U. Engberg and M Nielsen. A calculus of communicating systems with label-passing. Technical Report DAIMI PB-208, University of Aarhus, 1986.

[Eng77] R. Engelking. *General Topology.* PWN — Polish Scientific Publishers, 1977.

[Gro94] R. Grosu. *A Formal Foundation for Concurrent Object Oriented Programming.* PhD thesis, Technische Universität München, 1994. Also available as report TUM-I9444, Technische Universität München.

[GS95a] R. Grosu and K. Stølen. A denotational model for mobile many-to-many dataflow networks. Technical report, Technische Universität München, 1995.

[GS95b] R. Grosu and K. Stølen. A denotational model for mobile point-to-point dataflow networks. Technical Report SFB 342/14/95 A, Technische Universität München, 1995.

[HBS73] C. Hewitt, P. Bishop, and R. Steiger. A universal modular actor formalism for artificial intelligence. In *Proc. IJCAI'73*, pages 235–245, 1973.

[Jon89] B. Jonsson. A fully abstract trace model for dataflow networks. In *Proc. 16th Annual ACM Symposium on Principles of Programming Languages*, pages 155–165, 1989.

[Kah74] G. Kahn. The semantics of a simple language for parallel programming. In *Proc. Information Processing 74*, pages 471–475. North-Holland, 1974.

[Kok87] J. N. Kok. A fully abstract semantics for data flow nets. In *Proc. PARLE'87, Lecture Notes in Computer Science 259*, pages 351–368, 1987.

[Mes91] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. Technical Report SRI-CSL-91-05, SRI, 1991.

[Mil92] R. Milner. Action structures. Technical Report ECS-LFCS-92-249, University of Edinburgh, 1992.

[MPS86] D. MacQueen, G. Plotkin, and R. Sethi. An Ideal Model for Recursive Polymorphic Types. *Information and Control*, 71:95–130, 1986.

[MPW92a] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part I. *Information and Computation*, 100:1–40, 1992.

[MPW92b] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part II. *Information and Computation*, 100:41–77, 1992.

[Niv82] M Nivat. Behaviours of processes and synchronized systems of processes. In *Proc. Theoretical Foundations of Programming Methodology*, pages 473–551, 1982.

[Par83] D. Park. The "fairness" problem and nondeterministic computing networks. In *Proc. 4th Foundations of Computer Science, Mathematical Centre Tracts 159*, pages 133–161. Mathematisch Centrum Amsterdam, 1983.

[Rus90] J. R. Russell. On oraclizable networks and Kahn's principle. In *Proc. POPL'90*, pages 320–328, 1990.

[SKL90] V. A. Saraswat, K. Kahn, and J. Levy. Janus: A step towards distributed constraint programming. In *Proc. North American Conference on Logic Programming*, 1990.

[SRP91] V. A. Saraswat, M. Rinard, and P Panangaden. Semantic foundations of concurrent constraint programming. In *Proc. POPL'91*, pages 333–352, 1991.

[Sut75] W.A. Sutherland. *Introduction to metric and topological spaces.* Claredon Press - Oxford, 1975.

[Tho89] B. Thomsen. A calculus of higher order communicating systems. In *Proc. POPL'89*, 1989.

A Streams and Named Stream Tuples

A stream is a finite or infinite sequence of elements. For any set of elements E, we use E^* to denote the set of all finite streams over E, and $[E]$ to denote the set of all infinite streams over E. For any infinite stream s, we use $s{\downarrow_j}$ to denote the prefix of s containing exactly j elements. We use ϵ to denote the empty stream.

We define the metric of infinite streams generically with respect to an arbitrary discrete metric (E, ρ).

Definition 10 (The metric of infinite streams) *The metric of infinite streams* $([E], d)$ *over a discrete metric* (E, ρ) *is defined as follows:*

$[E] = \prod_{i \in \mathbb{N}} E$

$d(s,t) = inf\{2^{-j} \mid s{\downarrow_j} = t{\downarrow_j}\}$

□

This metric is also known as the Baire metric [Eng77].

Theorem 6 *The metric space of streams* $([E], d)$ *is complete.*

Proof sketch: See for example [Eng77]. □

A *named stream tuple* is a mapping $\theta \in (I \to [E])$ from a set of names to infinite streams. $\downarrow$ is overloaded to named stream tuples in a point-wise style, i.e. $\theta{\downarrow_j}$ denotes the result of applying $\downarrow_j$ to each component of θ.

Definition 11 (The metric of named stream tuples) *The metric of* named stream tuples $(I \to [E], d)$ *with names in* I *and elements in* (E, ρ) *is defined as follows:*

$I \to [E]$ *is the set of functions from the countable set* I *to the metric* $[E]$,

$d(s,t) = inf\{2^{-j} \mid s{\downarrow_j} = t{\downarrow_j}\}$

□

Theorem 7 *The metric space of named stream tuples* $(I \to [E], d)$ *is complete.*

Proof sketch: This metric is equivalent to the Cartesian product metric $\prod_{i \in I}[E]$ which is complete because $[E]$ is [Eng77]. □

B Metric Space Definitions

B.1 Metric Space Basics

The fundamental concept in metric spaces is the concept of distance.

Definition 12 (Metric Space) *A* metric space *is a pair* (D, d) *consisting of a nonempty set* D *and a mapping* $d \in D \times D \to R$*, called a* metric *or* distance*, which has the following properties:*

(1) $\forall x, y \in D: \quad d(x,y) = 0 \quad \Leftrightarrow \quad x = y$

(2) $\forall x, y \in D: \quad d(x,y) = d(y,x)$

(3) $\forall x, y, z \in D: \quad d(x,y) \leq d(x,z) + d(z,y)$ □

A very simple example of a metric is the discrete metric.

Definition 13 (The discrete metric) *The* discrete metric (D, d) *over a set* D *is defined as follows:*

$$d(x,y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$$

□

Measuring the distance between the elements of a sequence $(x_i)_{i\in\mathbb{N}}$ in D we obtain the familiar definitions for convergence and limits.

Definition 14 (Convergence and limits) *Let* (D, d) *be a metric space and let* $(x_i)_{i\in\mathbb{N}}$ *be a sequence in* D*.*

(1) *We say that* $(x_i)_{i\in\mathbb{N}}$ *is a* Cauchy sequence *whenever we have:*

$\forall \epsilon > 0 : \exists N \in \mathbb{N} : \forall n, m > N : d(x_n, x_m) < \epsilon.$

(2) *We say that* $(x_i)_{i\in\mathbb{N}}$ converges to $x \in D$ *denoted by* $x = lim_{n\to\infty} x_i$ *and call* x *the* limit *of* $(x_i)_{i\in\mathbb{N}}$ *whenever we have:*

$\forall \epsilon > 0 : \exists N \in \mathbb{N} : \forall n > N : d(x_n, x) < \epsilon.$

(3) *The metric space* (D, d) *is called* complete *whenever each Cauchy sequence converges to an element of* D*.*

□

Theorem 8 *The discrete metric is complete.*

Proof sketch: Each Cauchy sequence is constant from a given N. □

A very important class of functions over metric spaces is the class of *Lipschitz functions*.

Definition 15 (Lipschitz functions) *Let* (D_1, d_1) *and* (D_2, d_2) *be metric spaces and let* $f \in D_1 \to D_2$ *be a function. We call* f Lipschitz function *with constant* c *if there is a constant* $c \geq 0$ *such that the following condition is satisfied:*

$$d(f(x), f(y)) \leq c \cdot d(x, y)$$

For a function f *with arity* n *the above condition generalizes to:*

$$d(f(x_1, \ldots, x_n), f(y_1, \ldots, y_n)) \leq c \cdot max\{d(x_i, y_i) \mid i \in [1..n]\}$$

If $c = 1$ *we call* f non-expansive. *If* $c < 1$ *we call* f contractive. □

Theorem 9 *The composition of two Lipschitz functions* $f \in D_1 \to D_2$ *and* $g \in D_2 \to D_3$ *is a Lipschitz function with constant* $c_1 \cdot c_2$*.*

Proof sketch: $d(g(f(x_1)), g(f(x_2)) \leq c_2 \cdot d(f(x_1), f(x_2)) \leq c_2 \cdot c_1 \cdot d(x_1, x_2)$ □

Lemma 1 *The composition of a contractive and a non-expansive function is contractive. The composition of two non-expansive functions is non-expansive. Identity is non-expansive.* □

The main tool for handling recursion in metric spaces is the Banach's fix-point theorem. It guarantees the existence of a unique fix-point for every contractive function.

Theorem 10 (Banach's fix-point theorem) *Let (D, d) be a complete metric space and $f \in D \to D$ a contractive function. Then there exists an $x \in D$, such that the following holds:*

(1) $x = f(x)$ *(x is a fix-point of f)*
(2) $\forall y \in D : \; y = f(y) \Rightarrow y = x$ *(x is unique)*
(3) $\forall z \in D : \; x = lim_{n \to \infty} f^n(z)$ *where*
$f^0(z) = z$
$f^{n+1}(z) = f(f^n(z))$

Proof sketch: See [Eng77] or [Sut75]. □

Usually we want to use a parameterized version of this theorem.

Definition 16 (Parameterized fix-point) *Let $f \in D \times D_1 \times \ldots \times D_n \to D$ be a function of non-empty complete metric spaces that is contractive in its first argument. We define the* parameterized fix-point *function μf as follows:*

$(\mu f) \in D_1 \times \ldots \times D_n \to D$
$(\mu f)(y_1, \ldots, y_n) = x$

where x is the unique element of D such that $x = f(x, y_1, \ldots, y_n)$ as guaranteed by Banach's fix-point theorem. □

Theorem 11 *If f is contractive (non-expansive) so is μf.*

Proof sketch: See for example [MPS86] pages 114–115. □

C Full Abstraction

The context free syntax of a data-flow network is given by

$net ::= n \mid net \times net$

where n is a basic component. Network terms defined in this way are also called *rough terms*, because they do not contain information about their syntactic interface (I, O, P). By adding this interface information and the formation constraints of Section 5, we obtain the *context sensitive syntax*.

A context $C(.) :: (N \xrightarrow{\cup} [M^*]) \overset{I,O,P}{\rightarrow} (N \xrightarrow{\cup} [M^*])$ is a network with exactly one hole of type $(N \xrightarrow{\cup} [M^*]) \overset{I,O,P}{\rightarrow} (N \xrightarrow{\cup} [M^*])$. A network of this type is allowed to be substituted for the hole.

Given that each basic component can be understood as a mobile component as defined above, then the semantics of a network expression follows straightforwardly from the definition in Section 5. Let us denote this semantics by $\mathcal{D}$. We define the observation semantics of a network as the corresponding input/output behavior:

$$\mathcal{O}[n] \quad = \quad \{(x, y) \mid \exists f \in \mathcal{D}[n] : \ y = f(x)\}$$

Now, we define full abstraction as in [Jon89].

Definition 17 (Full abstraction) *The model $\mathcal{D}$ is said to be fully abstract with respect to $\mathcal{O}$ if for all networks $n, m :: (N \xrightarrow{\cup} [M^*]) \overset{I,O,P}{\rightarrow} (N \xrightarrow{\cup} [M^*])$*

$$\begin{array}{lll}
(1) & \mathcal{D}[n] = \mathcal{D}[m] \quad \Rightarrow \quad \mathcal{O}[n] = \mathcal{O}[n] & \{\mathcal{D} \textit{ is more distinguishing}\} \\
(2) & \forall C(.) : \mathcal{D}[n] = \mathcal{D}[m] \quad \Rightarrow \quad \mathcal{D}[C(n)] = \mathcal{D}[C(m)] & \{\mathcal{D} \textit{ is compositional}\} \\
(3) & \mathcal{D}[n] \neq \mathcal{D}[m] \quad \Rightarrow \quad \exists C(.) : \mathcal{O}[C(n)] \neq \mathcal{O}[C(m)] &
\end{array}$$

□

Theorem 12 *$\mathcal{D}$ is fully abstract with respect to $\mathcal{O}$.*

Proof sketch: Property (1) follows from the definition of $\mathcal{O}$. Property (2) follows by construction. To prove Property (3), suppose $\mathcal{D}[n] \neq \mathcal{D}[m]$. This means, there is an f which is contained in the first set, but not in the second. Since $\mathcal{D}[m]$ is closed, there is a θ such that $\forall g \in \mathcal{D}[m] : \ f(\theta) \neq g(\theta)$. Otherwise, closedness would require that f is also in $\mathcal{D}[m]$. As a consequence $(\theta, f(\theta)) \notin \mathcal{O}[m]$. Hence, $\mathcal{O}[n] \neq \mathcal{O}[m]$ □

Note that the full abstractness property is lost if the components are not required to be closed.

D Proof

We now prove that $F_1 \otimes F_2$ is a mobile component. The proof is split into several steps. Let

$$\begin{array}{l}
F_1 \hat{\otimes} F_2 = \{f \in (N \to [M^*]) \rightharpoonup (N \to [M^*]) \ \mid \ \forall \theta : \exists f_1 \in F_1, f_2 \in F_2 : \\
\qquad\qquad f(\theta) = \mathsf{rng}_{I,O,P}(\theta, \varphi \leftrightarrows \psi) \ \ \mathsf{where} \\
\qquad\qquad\quad \varphi = f_1(\vartheta \leftrightarrows \psi), \quad \psi = f_2(\vartheta \leftrightarrows \varphi), \quad \vartheta = \mathsf{dom}_{I,O,P}(\theta, \varphi \leftrightarrows \psi)\}
\end{array}$$

Lemma 1 $F_1 \hat{\otimes} F_2 \neq \emptyset$.

Proof: Since F_1 and F_2 are mobile components we may find functions f_1, f_2 such that $f_1 \in F_1$ and $f_2 \in F_2$. Based on these functions we construct a function f which is strongly pulse-driven, generic and satisfies the recursive definition above.

Let

$g \in ((N \to [M^*]) \times (N \to [M^*])) \times (N \to [M^*]) \to (N \to [M^*]) \times (N \to [M^*])$

$g((\varphi, \psi), \theta) = (f_1(\vartheta \leftrightarrows \psi),\ f_2(\vartheta \leftrightarrows \varphi)) \quad \mathsf{where} \quad \vartheta = \mathsf{dom}_{I,O,P}(\theta, \varphi \leftrightarrows \psi)$.

Theorem 9 (in the appendix) and the way g is defined in terms of strongly and weakly pulse-driven functions imply that g is strongly pulse-driven. Thus $\mu\, g$ is well-defined, in which case Theorem 11 (in the appendix) implies that $\mu\, g$ is strongly pulse-driven. Let

$f \in (N \to [M^*]) \to (N \to [M^*])$

$f(\theta) = \mathsf{rng}_{I,O,P}(\theta, \varphi \leftrightarrows \psi) \quad \mathsf{where} \quad (\varphi, \psi) = (\mu\, g)(\theta)$.

Theorem 9 and the way f is defined in terms of strongly and weakly pulse-driven functions imply that f is strongly pulse-driven.

Finally, since $\exists f_1, f_2 : \forall \theta : P$ implies $\forall \theta : \exists f_1, f_2 : P$ it follows that $f \in F_1 \hat{\otimes} F_2$. □

Lemma 2 $F_1 \hat{\otimes} F_2$ *is generic.*

Proof: That f is generic is a consequence of the next two propositions.

Proposition 1 $f(\theta) = f(\mathsf{dom}_{I,O,P}(\theta, f(\theta)))$.

Proof: First, note that $f(\theta) = f(\mathsf{dom}_{I,O,P}(\theta, \varphi \leftrightarrows \psi))$ because

$\mathsf{rng}_{I,O,P}(\theta, \varphi \leftrightarrows \psi) = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, \varphi \leftrightarrows \psi), \varphi \leftrightarrows \psi)$ {by Theorem 4}

$\vartheta = \mathsf{dom}_{I,O,P}(\theta, \varphi \leftrightarrows \psi) = \mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, \varphi \leftrightarrows \psi), \varphi \leftrightarrows \psi)$ {by Theorem 4.}

Now

$f(\mathsf{dom}_{I,O,P}(\theta, f(\theta))) =$

$f(\mathsf{dom}_{I,O,P}(\theta, \mathsf{rng}_{I,O,P}(\theta, \varphi \leftrightarrows \psi))) =$ {by definition of f}

$f(\mathsf{dom}_{I,O,P}(\theta, \varphi \leftrightarrows \psi)) =$ {by Theorem 4}

$f(\theta)$ {by above remark}

□

Proposition 2 $f(\theta) = \mathsf{rng}_{I,O,P}(\theta, f(\theta))$.

Proof:

$\mathsf{rng}_{I,O,P}(\theta, f(\theta)) =$

$\mathsf{rng}_{I,O,P}(\theta, \mathsf{rng}_{I,O,P}(\theta, \varphi \leftrightarrows \psi)) =$ {by definition of f}

$\mathsf{rng}_{I,O,P}(\theta, \varphi \leftrightarrows \psi) =$ {by Theorem 4}

$f(\theta)$ {by definition of f}

□ □

Lemma 3 *If* $\theta \in N \xrightarrow{\mathsf{u}} [M^*]$ *then* $\leftrightarrows$ *can be replaced by* $+$ *in the definition of* $\hat{\otimes}$. *Moreover* $\varphi \leftrightarrows \psi, \vartheta \leftrightarrows \psi, \vartheta \leftrightarrows \varphi \in N \xrightarrow{\mathsf{u}} [M^*]$.

Proof: Let

$$\begin{array}{llll} \mathsf{ep}^1_n & = \mathsf{ep}_{I_1,O_1,P_1}(\vartheta \leftrightarrow\!\!\!\!\!\!\rightarrow \psi, \varphi)(n), & \mathsf{pp}^1_n & = \mathsf{pp}_{I_1,O_1,P_1}(\vartheta \leftrightarrow\!\!\!\!\!\!\rightarrow \psi, \varphi)(n), \\ \mathsf{ep}^2_n & = \mathsf{ep}_{I_2,O_2,P_2}(\vartheta \leftrightarrow\!\!\!\!\!\!\rightarrow \varphi, \psi)(n), & \mathsf{pp}^2_n & = \mathsf{pp}_{I_2,O_2,P_2}(\vartheta \leftrightarrow\!\!\!\!\!\!\rightarrow \varphi, \psi)(n), \\ \mathsf{ep}_n & = \mathsf{ep}_{I,O,P}(\theta, \varphi \leftrightarrow\!\!\!\!\!\!\rightarrow \psi)(n), & \mathsf{pp}_n & = \mathsf{pp}_{I,O,P}(\theta, \varphi \leftrightarrow\!\!\!\!\!\!\rightarrow \psi)(n). \end{array}$$

Proposition 3 *If* $\theta \in N \xrightarrow{\mathsf{u}} [M^*]$ *then* $\varphi, \psi, \vartheta \leftrightarrow\!\!\!\!\!\!\rightarrow \psi, \vartheta \leftrightarrow\!\!\!\!\!\!\rightarrow \varphi \in N \xrightarrow{\mathsf{u}} [M^*]$ *and for all* n

$$\begin{array}{llllll} \mathsf{ep}^1_n \cap \mathsf{pp}^1_n & = & \mathsf{ep}^2_n \cap \mathsf{pp}^2_n & = & \mathsf{ep}_n \cap \mathsf{pp}_n & = & \emptyset \\ (\mathsf{pp}^1_n \cup \mathsf{ep}^1_n) & \cap & (\mathsf{pp}^2_n \cup \mathsf{ep}^2_n) & = & \emptyset, & & \end{array}$$

$$\begin{array}{lllll} \mathsf{ep}_n & = (\mathsf{ep}^1_n \setminus \widetilde{\mathsf{ep}^2_n}) & \cup & (\mathsf{ep}^2_n \setminus \widetilde{\mathsf{ep}^1_n}) & \\ \mathsf{pp}_n & = (\mathsf{ep}^1_n \cap \widetilde{\mathsf{ep}^2_n}) & \cup & (\mathsf{ep}^2_n \cap \widetilde{\mathsf{ep}^1_n}) & \cup \quad \mathsf{pp}^1_n \cup \mathsf{pp}^2_n \end{array}$$

Proof: Supose the above equalities are our induction hypothesis.

Base case:

$$\begin{array}{lll} \mathsf{pp}^1_1 = ?!P_1, & \mathsf{pp}^2_1 = ?!P_2, & \mathsf{pp}_1 = ?!(P_1 \cup P_2 \cup IO), \\ \mathsf{ep}^1_1 = ?I_1 \cup !O_1, & \mathsf{ep}^2_1 = ?I_2 \cup !O_2, & \mathsf{ep}_1 = ?I \cup !O, \\ \widetilde{\mathsf{ep}^1_1} = !I_1 \cup ?O_1, & \widetilde{\mathsf{ep}^2_1} = !I_2 \cup ?O_2. & \end{array}$$

These values clearly satisfy the above equations for $n = 1$. Together with the port-unicity preserving property of f_1 and f_2 they also assure the port unicity of $\varphi, \psi, \vartheta \leftrightarrow\!\!\!\!\!\!\rightarrow \psi, \vartheta \leftrightarrow\!\!\!\!\!\!\rightarrow \varphi$ and $\varphi \leftrightarrow\!\!\!\!\!\!\rightarrow \psi$ for $n = 1$.

Induction step: Expanding the definitions of ep and pp we obtain

$$\begin{array}{llll} \mathsf{ep}^1_{n+1} & = (\mathsf{ep}^1_n \cup r^1_n \cup g^1_n) \setminus (s^1_n \cup h^1_n), & \mathsf{pp}^1_{n+1} & = (\mathsf{pp}^1_n \cup h^1_n) \setminus (s^1_n \cup \widetilde{s^1_n}), \\ \mathsf{ep}^2_{n+1} & = (\mathsf{ep}^2_n \cup r^2_n \cup g^2_n) \setminus (s^2_n \cup h^2_n), & \mathsf{pp}^2_{n+1} & = (\mathsf{pp}^2_n \cup h^2_n) \setminus (s^2_n \cup \widetilde{s^2_n}), \\ \mathsf{ep}_{n+1} & = (\mathsf{ep}_n \cup r_n \cup g_n) \setminus (s_n \cup h_n), & \mathsf{pp}_{n+1} & = (\mathsf{pp}_n \cup h_n) \setminus (s_n \cup \widetilde{s_n}) \end{array}$$

where

$$\begin{array}{llll} r^1_n & = \bigcup_{?i \in \mathsf{ep}^1_n} \{c \mid c \in \overline{\mathsf{pp}^1_n \cup \mathsf{ep}^1_n} \cap (\vartheta \leftrightarrow\!\!\!\!\!\!\rightarrow \psi)(i)(n)\}, & h^1_n & = \{c, \widetilde{c} \mid c \in r^1_n \wedge \widetilde{c} \in \mathsf{ep}^1_n\} \\ r^2_n & = \bigcup_{?i \in \mathsf{ep}^2_n} \{c \mid c \in \overline{\mathsf{pp}^2_n \cup \mathsf{ep}^2_n} \cap (\vartheta \leftrightarrow\!\!\!\!\!\!\rightarrow \varphi)(i)(n)\}, & h^2_n & = \{c, \widetilde{c} \mid c \in r^2_n \wedge \widetilde{c} \in \mathsf{ep}^2_n\} \\ r_n & = \bigcup_{?i \in \mathsf{ep}_n} \{c \mid c \in \overline{\mathsf{pp}_n \cup \mathsf{ep}_n} \cap \theta(i)(n)\}, & h_n & = \{c, \widetilde{c} \mid c \in r_n \wedge \widetilde{c} \in \mathsf{ep}_n\} \end{array}$$

$$\begin{array}{llll} s^1_n & = \bigcup_{!i \in \mathsf{ep}^1_n} \{c \mid c \in (\mathsf{pp}^1_n \cup \mathsf{ep}^1_n) \cap \varphi(i)(n)\}), & g^1_n & = \{\widetilde{c} \mid c \in s^1_n \wedge c \in \mathsf{pp}^1_n\} \\ s^2_n & = \bigcup_{!i \in \mathsf{ep}^2_n} \{c \mid c \in (\mathsf{pp}^2_n \cup \mathsf{ep}^2_n) \cap \psi(i)(n)\}, & g^2_n & = \{\widetilde{c} \mid c \in s^2_n \wedge c \in \mathsf{pp}^2_n\} \\ s_n & = \bigcup_{!i \in \mathsf{ep}_n} \{c \mid c \in (\mathsf{pp}_n \cup \mathsf{ep}_n) \cap (\varphi \leftrightarrow\!\!\!\!\!\!\rightarrow \psi)(i)(n)\}, & g_n & = \{\widetilde{c} \mid c \in s_n \wedge c \in \mathsf{pp}_n\} \end{array}$$

We do now a case analysis which corresponds to each term of the above expressions.

1. External Input: $?i \in \mathsf{ep}^1_n \cap \mathsf{ep}_n$

By the induction hypothesis $?i \notin \widetilde{\mathsf{ep}^2_n}$. As a consequence $(\vartheta \leftrightarrow\!\!\!\!\!\!\rightarrow \psi)(i)(n) = \vartheta(i)(n)$. Suppose $c \in \vartheta(i)(n)$. Clearly $c \in \overline{\mathsf{ep}_n \cup \mathsf{pp}_n}$. There are two cases $\widetilde{c} \notin \mathsf{ep}^1_n$ and $\widetilde{c} \in \mathsf{ep}^1_n$:

$\underline{\widetilde{c} \notin \mathsf{ep}^1_n}$

$$\begin{array}{ll} c \in r^1_n, & \{\text{by definition}\} \\ c \notin h^1_n, & \{\text{by definition}\} \\ c \notin s^1_n, & \{(\mathsf{ep}^1_n \cup \mathsf{pp}^1_n) \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\} \end{array}$$

Hence $c \in \mathsf{ep}^1_{n+1}$ and $c \notin \mathsf{pp}^1_{n+1}$.

$c \notin \mathsf{ep}^2_n$, $\{\mathsf{ep}^2_n \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$
$c \notin r^2_n$, $\{u(\theta),\ (\mathsf{ep}^1_n \cup \mathsf{pp}^1_n) \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$
$c \notin s^2_n$, $\{(\mathsf{ep}^2_n \cup \mathsf{pp}^2_n) \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$

Hence $c \notin \mathsf{ep}^2_{n+1}$ and $c \notin \mathsf{pp}^2_{n+1}$.

$c \in r_n$, {by definition}
$c \notin h_n$, $\{a.\ \widetilde{c} \notin \mathsf{ep}^2_n\}$
$c \in h_n$, $\{b.\ \widetilde{c} \in \mathsf{ep}^2_n\}$
$c \notin s_n$, $\{(\mathsf{ep}_n \cup \mathsf{pp}_n) \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$

Hence $c \in \mathsf{ep}_{n+1}$ and $c \notin \mathsf{pp}_{n+1}$ in case a and $c \notin \mathsf{ep}_{n+1}$ and $c \in \mathsf{pp}_{n+1}$ in case b.

$\underline{\widetilde{c} \in \mathsf{ep}^1_n}$

$c \in r^1_n$ {by definition}
$c \in h^1_n$ {by definition}
$c \notin s^1_n$ $\{(\mathsf{ep}^1_n \cup \mathsf{pp}^1_n) \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$

Hence $c \notin \mathsf{ep}^1_{n+1}$ and $c \in \mathsf{pp}^1_{n+1}$. For f_2 nothing changes. Hence $c \notin \mathsf{ep}^2_{n+1}$ and $c \notin \mathsf{pp}^2_{n+1}$.

$c \in r_n$ {by definition}
$c \in h_n$ {by definition}
$c \notin s^1_n$ $\{(\mathsf{ep}_n \cup \mathsf{pp}_n) \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$

Hence $c \notin \mathsf{ep}_{n+1}$ and $c \in \mathsf{pp}_{n+1}$.

In both cases all the above equations are satisfied. Moreover, the port-unicity of $\vartheta \nleftrightarrow \psi$ and $\vartheta \nleftrightarrow \varphi$ at time n, and the port-unicity preserving property of f_1 and f_2 implies the port-unicity of φ and ψ at time $n+1$. Then the above disjointness equations imply the unicity of $\vartheta \nleftrightarrow \psi$, $\vartheta \nleftrightarrow \varphi$ and $\varphi \nleftrightarrow \psi$ at time $n+1$.

2. Internal Input: $?i \in \mathsf{ep}^1_n \cap \widetilde{\mathsf{ep}^2_n}$

By induction hypothesis $?i \notin \mathsf{ep}_n$. Hence $(\vartheta \nleftrightarrow \psi)(i)(n) = \psi(i)(n)$. Suppose $c \in \psi(i)(n)$. Then there are two disjoint cases: $c \in \mathsf{ep}^2_n$ or $c \in \mathsf{pp}^2_n$.

$\underline{c \in \mathsf{ep}^2_n}$

$c \in r^1_n$, {by definition}
$c \notin h^1_n$ $\{a.\ \widetilde{c} \notin \mathsf{ep}^1_n\}$
$c \in h^1_n$ $\{b.\ \widetilde{c} \in \mathsf{ep}^1_n\}$
$c \notin s^1_n$, $\{(\mathsf{ep}^1_n \cup \mathsf{pp}^1_n) \cap \mathsf{ep}^2_n = \emptyset\}$

Hence $c \in \mathsf{ep}^1_{n+1}$ and $c \notin \mathsf{pp}^1_{n+1}$ in case a or $c \notin \mathsf{ep}^1_{n+1}$ and $c \in \mathsf{pp}^1_{n+1}$ in case b.

$c \notin r^2_n$, $\{\mathsf{ep}^2_n \cap (\mathsf{ep}^1_n \cup \mathsf{pp}^1_n) = \mathsf{ep}^2_n \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$
$c \in s^2_n$, {by definition}
$c \notin g^2_n$, $\{\mathsf{ep}^2_n \cap \mathsf{pp}^2_n = \emptyset\}$

Hence $c \notin \mathsf{ep}^2_{n+1}$ and $c \notin \mathsf{pp}^2_{n+1}$.

$c \in \mathsf{ep}_n \wedge c \notin \mathsf{pp}_n$, $\{a.\ \widetilde{c} \notin \mathsf{ep}^1_n\}$
$c \notin \mathsf{ep}_n \wedge c \in \mathsf{pp}_n$ $\{b.\ \widetilde{c} \in \mathsf{ep}^1_n\}$
$c \notin r_n$, $\{\mathsf{ep}^2_n \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$
$c \notin s_n$, $\{u(\psi),\ (\mathsf{ep}^1_n \cup \mathsf{pp}^1_n) \cap \mathsf{ep}^2_n = \emptyset\}$

Hence $c \in \mathsf{ep}_{n+1}$ and $c \notin \mathsf{pp}_{n+1}$ in case a or $c \notin \mathsf{ep}_{n+1}$ and $c \in \mathsf{pp}_{n+1}$ in case b.

$\underline{c \in \mathsf{pp}^2_n}$

$c \in r^1_n$, {by definition}
$c \notin h^1_n$ $\{c, \widetilde{c} \notin \mathsf{ep}^1_n\}$
$c \notin s^1_n$, $\{(\mathsf{ep}^1_n \cup \mathsf{pp}^1_n) \cap \mathsf{pp}^2_n = \emptyset\}$

Hence $c \in \mathsf{ep}^1_{n+1}$ and $c \notin \mathsf{pp}^1_{n+1}$.

$c \notin r^2_n$, $\{\mathsf{pp}^2_n \cap (\mathsf{ep}^1_n \cup \mathsf{pp}^1_n) = \mathsf{pp}^2_n \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$
$c \in s^2_n$, {by definition}
$\widetilde{c} \in g^2_n$, {by definition}

Hence $c \notin \mathsf{ep}^2_{n+1}$ and $c \notin \mathsf{pp}^2_{n+1}$ and $\widetilde{c} \in \mathsf{ep}^2_n$.

$c \notin \mathsf{ep}_n$ $\{\mathsf{pp}^2_n \cap \mathsf{ep}_n = \emptyset\}$
$c \in \mathsf{pp}_n$ {induction hypothesis}
$c \notin r_n$, $\{\mathsf{pp}^2_n \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$
$c \notin s_n$, $\{u(\psi),\ (\mathsf{ep}^1_n \cup \mathsf{pp}^1_n) \cap \mathsf{pp}^2_n = \emptyset\}$

Hence $c \notin \mathsf{ep}_{n+1}$ and $c \in \mathsf{pp}_{n+1}$.

It is easy to show that all the above equations hold. Moreover, a similar argument as before, proves the port-unicity.

3. External Output: $!i \in \mathsf{ep}^1_n \cap \mathsf{ep}_n$

By the induction hypothesis $!i \notin \widetilde{\mathsf{ep}^2_n}$. The only interesting case is if $c \in \varphi(i)(n)$ is private and its complement is not sent, i.e., if $c \in \mathsf{pp}^1_n$ and $\widetilde{c} \notin \varphi(i)(n)$.

$c \in s^1_n$, {by definition}
$\widetilde{c} \in g^1_n$, {by definition}

Hence $\widetilde{c} \in \mathsf{ep}^1_{n+1}$ and $c, \widetilde{c} \notin \mathsf{pp}^1_{n+1}$.

$c \notin \mathsf{ep}^2_n$ $\{\mathsf{pp}^1_n \cap \mathsf{ep}^2_n = \emptyset\}$
$c, \widetilde{c} \notin r^2_n$, $\{u(\varphi),\ \mathsf{pp}^1_n \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$
$c, \widetilde{c} \notin s^2_n$, $\{\mathsf{pp}^1_n \cap (\mathsf{ep}^2_n \cup \mathsf{pp}^2_n) = \emptyset\}$

Hence $c, \widetilde{c} \notin \mathsf{ep}^2_{n+1}$ and $c, \widetilde{c} \notin \mathsf{pp}^2_{n+1}$.

$c \in s_n$, {by definition}
$\widetilde{c} \in g_n$, {by definition}

Hence $\widetilde{c} \in \mathsf{ep}_{n+1}$ and $c, \widetilde{c} \notin \mathsf{pp}_{n+1}$.

It is easy to show that all the above equations hold. Moreover, a similar argument as before, proves the port-unicity.

4. Internal Output: $!i \in \mathsf{ep}^1_n \cap \widetilde{\mathsf{ep}^2_n}$

By the induction hypothesis $!i \notin \mathsf{ep}_n$. The only interesting case is if $c \in \varphi(i)(n)$ is private and its complement is not sent, i.e., if $c \in \mathsf{pp}^1_n$ and $\widetilde{c} \notin \varphi(i)(n)$.

$c \in s^1_n$, {by definition}
$\widetilde{c} \in g^1_n$, {by definition}

Hence $\widetilde{c} \in \mathsf{ep}^1_{n+1}$, $c \notin \mathsf{ep}^1_{n+1}$ and $c, \widetilde{c} \notin \mathsf{pp}^1_{n+1}$.

$c \in r^2_n$, {by definition}
$\widetilde{c} \notin r^2_n$, $\{u(\varphi),\ \mathsf{pp}^1_n \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$
$\widetilde{c} \notin s^2_n$, $\{\mathsf{pp}^1_n \cap (\mathsf{ep}^2_n \cup \mathsf{pp}^2_n) = \emptyset\}$

Hence $c \in \mathsf{ep}^2_{n+1}$, $\widetilde{c} \notin \mathsf{ep}^2_{n+1}$ and $c, \widetilde{c} \notin \mathsf{pp}^2_{n+1}$.

$c \notin \mathsf{ep}_n$ $\{\mathsf{pp}^1_n \cap \mathsf{ep}_n = \emptyset\}$
$c \in \mathsf{pp}_n$ {induction hypothesis}
$c \notin r_n$, $\{\mathsf{pp}^1_n \cap \overline{\mathsf{ep}_n \cup \mathsf{pp}_n} = \emptyset\}$
$c \notin s_n$, $\{u(\psi),\ (\mathsf{ep}^2_n \cup \mathsf{pp}^2_n) \cap \mathsf{pp}^1_n = \emptyset\}$

Hence $c \notin \mathsf{ep}_{n+1}$ and $c \in \mathsf{pp}_{n+1}$.

It is easy to show that all the above equations hold. Moreover, a similar argument as before, proves the port-unicity. This also completes all the cases for f_1. A similar argument applies for f_2. □

As a consequence of the above proposition, ϑ, φ and ψ have disjoint domains, are port unique and contain disjoint sets of ports. Consequently, the total sums $\vartheta \leftrightarrows \psi, \vartheta \leftrightarrows \varphi$ and $\varphi \leftrightarrows \psi$ are port unique and

$$\vartheta \leftrightarrows \psi = \vartheta + \psi, \quad \vartheta \leftrightarrows \varphi = \vartheta + \varphi, \quad \varphi \leftrightarrows \psi = \varphi + \psi$$ □

Lemma 4 *$F_1 \otimes F_2$ is closed.*

Proof: To see that $F_1 \otimes F_2$ is closed, let $f \in (N \xrightarrow{\mathsf{u}} [M^*])\ ^{I,O,P}\!\!\rightharpoonup\ (N \xrightarrow{\mathsf{u}} [M^*])$, and assume that

$$\forall \theta : \exists f' \in F_1 \otimes F_2 :\ f(\theta) = f'(\theta).$$

The definition of $\otimes$ implies that for any θ there are $f_1 \in F_1$, $f_2 \in F_2$ such that:

$f(\theta) = \mathsf{rng}_{I,O,P}(\theta, \varphi + \psi)$ where
$\varphi = f_1(\vartheta + \psi),\ \ \psi = f_2(\vartheta + \varphi),\ \ \vartheta = \mathsf{dom}_{I,O,P}(\theta, \varphi + \psi)$

By the definition of $\otimes$, it follows that $f \in F_1 \otimes F_2$. □

As a consequence of the above lemmas $F_1 \otimes F_2$ is a mobile component.

Theorem 13 *$F_1 \otimes F_2 \subseteq (N \xrightarrow{\mathsf{u}} [M^*])\ ^{I,O,P}\!\!\rightharpoonup\ (N \xrightarrow{\mathsf{u}} [M^*])$ is a mobile component.*

Proof: A consequence of Lemmas 1, 4, 2 and 3. □

SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

bisher erschienen :

Reihe A

342/1/90 A Robert Gold, Walter Vogler: Quality Criteria for Partial Order Semantics of Place/Transition-Nets, Januar 1990

342/2/90 A Reinhard Fößmeier: Die Rolle der Lastverteilung bei der numerischen Parallelprogrammierung, Februar 1990

342/3/90 A Klaus-Jörn Lange, Peter Rossmanith: Two Results on Unambiguous Circuits, Februar 1990

342/4/90 A Michael Griebel: Zur Lösung von Finite-Differenzen- und Finite-Element-Gleichungen mittels der Hierarchischen Transformations-Mehrgitter-Methode

342/5/90 A Reinhold Letz, Johann Schumann, Stephan Bayerl, Wolfgang Bibel: SETHEO: A High-Performance Theorem Prover

342/6/90 A Johann Schumann, Reinhold Letz: PARTHEO: A High Performance Parallel Theorem Prover

342/7/90 A Johann Schumann, Norbert Trapp, Martin van der Koelen: SETHEO/PARTHEO Users Manual

342/8/90 A Christian Suttner, Wolfgang Ertel: Using Connectionist Networks for Guiding the Search of a Theorem Prover

342/9/90 A Hans-Jörg Beier, Thomas Bemmerl, Arndt Bode, Hubert Ertl, Olav Hansen, Josef Haunerdinger, Paul Hofstetter, Jaroslav Kremenek, Robert Lindhof, Thomas Ludwig, Peter Luksch, Thomas Treml: TOPSYS, Tools for Parallel Systems (Artikelsammlung)

342/10/90 A Walter Vogler: Bisimulation and Action Refinement

342/11/90 A Jörg Desel, Javier Esparza: Reachability in Reversible Free- Choice Systems

342/12/90 A Rob van Glabbeek, Ursula Goltz: Equivalences and Refinement

342/13/90 A Rob van Glabbeek: The Linear Time - Branching Time Spectrum

342/14/90 A Johannes Bauer, Thomas Bemmerl, Thomas Treml: Leistungsanalyse von verteilten Beobachtungs- und Bewertungswerkzeugen

Reihe A

342/15/90 A Peter Rossmanith: The Owner Concept for PRAMs
342/16/90 A G. Böckle, S. Trosch: A Simulator for VLIW-Architectures
342/17/90 A P. Slavkovsky, U. Rüde: Schnellere Berechnung klassischer Matrix-Multiplikationen
342/18/90 A Christoph Zenger: SPARSE GRIDS
342/19/90 A Michael Griebel, Michael Schneider, Christoph Zenger: A combination technique for the solution of sparse grid problems
342/20/90 A Michael Griebel: A Parallelizable and Vectorizable Multi- Level-Algorithm on Sparse Grids
342/21/90 A V. Diekert, E. Ochmanski, K. Reinhardt: On confluent semi-commutations-decidability and complexity results
342/22/90 A Manfred Broy, Claus Dendorfer: Functional Modelling of Operating System Structures by Timed Higher Order Stream Processing Functions
342/23/90 A Rob van Glabbeek, Ursula Goltz: A Deadlock-sensitive Congruence for Action Refinement
342/24/90 A Manfred Broy: On the Design and Verification of a Simple Distributed Spanning Tree Algorithm
342/25/90 A Thomas Bemmerl, Arndt Bode, Peter Braun, Olav Hansen, Peter Luksch, Roland Wismüller: TOPSYS - Tools for Parallel Systems (User's Overview and User's Manuals)
342/26/90 A Thomas Bemmerl, Arndt Bode, Thomas Ludwig, Stefan Tritscher: MMK - Multiprocessor Multitasking Kernel (User's Guide and User's Reference Manual)
342/27/90 A Wolfgang Ertel: Random Competition: A Simple, but Efficient Method for Parallelizing Inference Systems
342/28/90 A Rob van Glabbeek, Frits Vaandrager: Modular Specification of Process Algebras
342/29/90 A Rob van Glabbeek, Peter Weijland: Branching Time and Abstraction in Bisimulation Semantics
342/30/90 A Michael Griebel: Parallel Multigrid Methods on Sparse Grids
342/31/90 A Rolf Niedermeier, Peter Rossmanith: Unambiguous Simulations of Auxiliary Pushdown Automata and Circuits
342/32/90 A Inga Niepel, Peter Rossmanith: Uniform Circuits and Exclusive Read PRAMs
342/33/90 A Dr. Hermann Hellwagner: A Survey of Virtually Shared Memory Schemes
342/1/91 A Walter Vogler: Is Partial Order Semantics Necessary for Action Refinement?
342/2/91 A Manfred Broy, Frank Dederichs, Claus Dendorfer, Rainer Weber: Characterizing the Behaviour of Reactive Systems by Trace Sets
342/3/91 A Ulrich Furbach, Christian Suttner, Bertram Fronhöfer: Massively Parallel Inference Systems

Reihe A

342/4/91 A Rudolf Bayer: Non-deterministic Computing, Transactions and Recursive Atomicity

342/5/91 A Robert Gold: Dataflow semantics for Petri nets

342/6/91 A A. Heise; C. Dimitrovici: Transformation und Komposition von P/T-Netzen unter Erhaltung wesentlicher Eigenschaften

342/7/91 A Walter Vogler: Asynchronous Communication of Petri Nets and the Refinement of Transitions

342/8/91 A Walter Vogler: Generalized OM-Bisimulation

342/9/91 A Christoph Zenger, Klaus Hallatschek: Fouriertransformation auf dünnen Gittern mit hierarchischen Basen

342/10/91 A Erwin Loibl, Hans Obermaier, Markus Pawlowski: Towards Parallelism in a Relational Database System

342/11/91 A Michael Werner: Implementierung von Algorithmen zur Kompaktifizierung von Programmen für VLIW-Architekturen

342/12/91 A Reiner Müller: Implementierung von Algorithmen zur Optimierung von Schleifen mit Hilfe von Software-Pipelining Techniken

342/13/91 A Sally Baker, Hans-Jörg Beier, Thomas Bemmerl, Arndt Bode, Hubert Ertl, Udo Graf, Olav Hansen, Josef Haunerdinger, Paul Hofstetter, Rainer Knödlseder, Jaroslav Kremenek, Siegfried Langenbuch, Robert Lindhof, Thomas Ludwig, Peter Luksch, Roy Milner, Bernhard Ries, Thomas Treml: TOPSYS - Tools for Parallel Systems (Artikelsammlung); 2., erweiterte Auflage

342/14/91 A Michael Griebel: The combination technique for the sparse grid solution of PDE's on multiprocessor machines

342/15/91 A Thomas F. Gritzner, Manfred Broy: A Link Between Process Algebras and Abstract Relation Algebras?

342/16/91 A Thomas Bemmerl, Arndt Bode, Peter Braun, Olav Hansen, Thomas Treml, Roland Wismüller: The Design and Implementation of TOPSYS

342/17/91 A Ulrich Furbach: Answers for disjunctive logic programs

342/18/91 A Ulrich Furbach: Splitting as a source of parallelism in disjunctive logic programs

342/19/91 A Gerhard W. Zumbusch: Adaptive parallele Multilevel-Methoden zur Lösung elliptischer Randwertprobleme

342/20/91 A M. Jobmann, J. Schumann: Modelling and Performance Analysis of a Parallel Theorem Prover

342/21/91 A Hans-Joachim Bungartz: An Adaptive Poisson Solver Using Hierarchical Bases and Sparse Grids

342/22/91 A Wolfgang Ertel, Theodor Gemenis, Johann M. Ph. Schumann, Christian B. Suttner, Rainer Weber, Zongyan Qiu: Formalisms and Languages for Specifying Parallel Inference Systems

342/23/91 A Astrid Kiehn: Local and Global Causes

Reihe A

342/24/91 A	Johann M.Ph. Schumann: Parallelization of Inference Systems by using an Abstract Machine
342/25/91 A	Eike Jessen: Speedup Analysis by Hierarchical Load Decomposition
342/26/91 A	Thomas F. Gritzner: A Simple Toy Example of a Distributed System: On the Design of a Connecting Switch
342/27/91 A	Thomas Schnekenburger, Andreas Weininger, Michael Friedrich: Introduction to the Parallel and Distributed Programming Language ParMod-C
342/28/91 A	Claus Dendorfer: Funktionale Modellierung eines Postsystems
342/29/91 A	Michael Griebel: Multilevel algorithms considered as iterative methods on indefinite systems
342/30/91 A	W. Reisig: Parallel Composition of Liveness
342/31/91 A	Thomas Bemmerl, Christian Kasperbauer, Martin Mairandres, Bernhard Ries: Programming Tools for Distributed Multiprocessor Computing Environments
342/32/91 A	Frank Leßke: On constructive specifications of abstract data types using temporal logic
342/1/92 A	L. Kanal, C.B. Suttner (Editors): Informal Proceedings of the Workshop on Parallel Processing for AI
342/2/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: The Design of Distributed Systems - An Introduction to FOCUS
342/2-2/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: The Design of Distributed Systems - An Introduction to FOCUS - Revised Version (erschienen im Januar 1993)
342/3/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: Summary of Case Studies in FOCUS - a Design Method for Distributed Systems
342/4/92 A	Claus Dendorfer, Rainer Weber: Development and Implementation of a Communication Protocol - An Exercise in FOCUS
342/5/92 A	Michael Friedrich: Sprachmittel und Werkzeuge zur Unterstützung paralleler und verteilter Programmierung
342/6/92 A	Thomas F. Gritzner: The Action Graph Model as a Link between Abstract Relation Algebras and Process-Algebraic Specifications
342/7/92 A	Sergei Gorlatch: Parallel Program Development for a Recursive Numerical Algorithm: a Case Study
342/8/92 A	Henning Spruth, Georg Sigl, Frank Johannes: Parallel Algorithms for Slicing Based Final Placement
342/9/92 A	Herbert Bauer, Christian Sporrer, Thomas Krodel: On Distributed Logic Simulation Using Time Warp
342/10/92 A	H. Bungartz, M. Griebel, U. Rüde: Extrapolation, Combination and Sparse Grid Techniques for Elliptic Boundary Value Problems

Reihe A

342/11/92 A M. Griebel, W. Huber, U. Rüde, T. Störtkuhl: The Combination Technique for Parallel Sparse-Grid-Preconditioning and -Solution of PDEs on Multiprocessor Machines and Workstation Networks

342/12/92 A Rolf Niedermeier, Peter Rossmanith: Optimal Parallel Algorithms for Computing Recursively Defined Functions

342/13/92 A Rainer Weber: Eine Methodik für die formale Anforderungsspezifkation verteilter Systeme

342/14/92 A Michael Griebel: Grid– and point–oriented multilevel algorithms

342/15/92 A M. Griebel, C. Zenger, S. Zimmer: Improved multilevel algorithms for full and sparse grid problems

342/16/92 A J. Desel, D. Gomm, E. Kindler, B. Paech, R. Walter: Bausteine eines kompositionalen Beweiskalküls für netzmodellierte Systeme

342/17/92 A Frank Dederichs: Transformation verteilter Systeme: Von applikativen zu prozeduralen Darstellungen

342/18/92 A Andreas Listl, Markus Pawlowski: Parallel Cache Management of a RDBMS

342/19/92 A Erwin Loibl, Markus Pawlowski, Christian Roth: PART: A Parallel Relational Toolbox as Basis for the Optimization and Interpretation of Parallel Queries

342/20/92 A Jörg Desel, Wolfgang Reisig: The Synthesis Problem of Petri Nets

342/21/92 A Robert Balder, Christoph Zenger: The d-dimensional Helmholtz equation on sparse Grids

342/22/92 A Ilko Michler: Neuronale Netzwerk-Paradigmen zum Erlernen von Heuristiken

342/23/92 A Wolfgang Reisig: Elements of a Temporal Logic. Coping with Concurrency

342/24/92 A T. Störtkuhl, Chr. Zenger, S. Zimmer: An asymptotic solution for the singularity at the angular point of the lid driven cavity

342/25/92 A Ekkart Kindler: Invariants, Compositionality and Substitution

342/26/92 A Thomas Bonk, Ulrich Rüde: Performance Analysis and Optimization of Numerically Intensive Programs

342/1/93 A M. Griebel, V. Thurner: The Efficient Solution of Fluid Dynamics Problems by the Combination Technique

342/2/93 A Ketil Stølen, Frank Dederichs, Rainer Weber: Assumption / Commitment Rules for Networks of Asynchronously Communicating Agents

342/3/93 A Thomas Schnekenburger: A Definition of Efficiency of Parallel Programs in Multi-Tasking Environments

342/4/93 A Hans-Joachim Bungartz, Michael Griebel, Dierk Röschke, Christoph Zenger: A Proof of Convergence for the Combination Technique for the Laplace Equation Using Tools of Symbolic Computation

Reihe A

342/5/93 A Manfred Kunde, Rolf Niedermeier, Peter Rossmanith: Faster Sorting and Routing on Grids with Diagonals

342/6/93 A Michael Griebel, Peter Oswald: Remarks on the Abstract Theory of Additive and Multiplicative Schwarz Algorithms

342/7/93 A Christian Sporrer, Herbert Bauer: Corolla Partitioning for Distributed Logic Simulation of VLSI Circuits

342/8/93 A Herbert Bauer, Christian Sporrer: Reducing Rollback Overhead in Time-Warp Based Distributed Simulation with Optimized Incremental State Saving

342/9/93 A Peter Slavkovsky: The Visibility Problem for Single-Valued Surface (z = f(x,y)): The Analysis and the Parallelization of Algorithms

342/10/93 A Ulrich Rüde: Multilevel, Extrapolation, and Sparse Grid Methods

342/11/93 A Hans Regler, Ulrich Rüde: Layout Optimization with Algebraic Multigrid Methods

342/12/93 A Dieter Barnard, Angelika Mader: Model Checking for the Modal Mu-Calculus using Gauß Elimination

342/13/93 A Christoph Pflaum, Ulrich Rüde: Gauß' Adaptive Relaxation for the Multilevel Solution of Partial Differential Equations on Sparse Grids

342/14/93 A Christoph Pflaum: Convergence of the Combination Technique for the Finite Element Solution of Poisson's Equation

342/15/93 A Michael Luby, Wolfgang Ertel: Optimal Parallelization of Las Vegas Algorithms

342/16/93 A Hans-Joachim Bungartz, Michael Griebel, Dierk Röschke, Christoph Zenger: Pointwise Convergence of the Combination Technique for Laplace's Equation

342/17/93 A Georg Stellner, Matthias Schumann, Stefan Lamberts, Thomas Ludwig, Arndt Bode, Martin Kiehl und Rainer Mehlhorn: Developing Multicomputer Applications on Networks of Workstations Using NXLib

342/18/93 A Max Fuchs, Ketil Stølen: Development of a Distributed Min/Max Component

342/19/93 A Johann K. Obermaier: Recovery and Transaction Management in Write-optimized Database Systems

342/20/93 A Sergej Gorlatch: Deriving Efficient Parallel Programs by Systemating Coarsing Specification Parallelism

342/01/94 A Reiner Hüttl, Michael Schneider: Parallel Adaptive Numerical Simulation

342/02/94 A Henning Spruth, Frank Johannes: Parallel Routing of VLSI Circuits Based on Net Independency

342/03/94 A Henning Spruth, Frank Johannes, Kurt Antreich: PHIroute: A Parallel Hierarchical Sea-of-Gates Router

Reihe A

342/04/94 A Martin Kiehl, Rainer Mehlhorn, Matthias Schumann: Parallel Multiple Shooting for Optimal Control Problems Under NX/2

342/05/94 A Christian Suttner, Christoph Goller, Peter Krauss, Klaus-Jörn Lange, Ludwig Thomas, Thomas Schnekenburger: Heuristic Optimization of Parallel Computations

342/06/94 A Andreas Listl: Using Subpages for Cache Coherency Control in Parallel Database Systems

342/07/94 A Manfred Broy, Ketil Stølen: Specification and Refinement of Finite Dataflow Networks - a Relational Approach

342/08/94 A Katharina Spies: Funktionale Spezifikation eines Kommunikationsprotokolls

342/09/94 A Peter A. Krauss: Applying a New Search Space Partitioning Method to Parallel Test Generation for Sequential Circuits

342/10/94 A Manfred Broy: A Functional Rephrasing of the Assumption/Commitment Specification Style

342/11/94 A Eckhardt Holz, Ketil Stølen: An Attempt to Embed a Restricted Version of SDL as a Target Language in Focus

342/12/94 A Christoph Pflaum: A Multi-Level-Algorithm for the Finite-Element-Solution of General Second Order Elliptic Differential Equations on Adaptive Sparse Grids

342/13/94 A Manfred Broy, Max Fuchs, Thomas F. Gritzner, Bernhard Schätz, Katharina Spies, Ketil Stølen: Summary of Case Studies in FOCUS - a Design Method for Distributed Systems

342/14/94 A Maximilian Fuchs: Technologieabhängigkeit von Spezifikationen digitaler Hardware

342/15/94 A M. Griebel, P. Oswald: Tensor Product Type Subspace Splittings And Multilevel Iterative Methods For Anisotropic Problems

342/16/94 A Gheorghe Ştefănescu: Algebra of Flownomials

342/17/94 A Ketil Stølen: A Refinement Relation Supporting the Transition from Unbounded to Bounded Communication Buffers

342/18/94 A Michael Griebel, Tilman Neuhoeffer: A Domain-Oriented Multilevel Algorithm-Implementation and Parallelization

342/19/94 A Michael Griebel, Walter Huber: Turbulence Simulation on Sparse Grids Using the Combination Method

342/20/94 A Johann Schumann: Using the Theorem Prover SETHEO for verifying the development of a Communication Protocol in FOCUS - A Case Study -

342/01/95 A Hans-Joachim Bungartz: Higher Order Finite Elements on Sparse Grids

342/02/95 A Tao Zhang, Seonglim Kang, Lester R. Lipsky: The Performance of Parallel Computers: Order Statistics and Amdahl's Law

342/03/95 A Lester R. Lipsky, Appie van de Liefvoort: Transformation of the Kronecker Product of Identical Servers to a Reduced Product Space

Reihe A

342/04/95 A Pierre Fiorini, Lester R. Lipsky, Wen-Jung Hsin, Appie van de Liefvoort: Auto-Correlation of Lag-k For Customers Departing From Semi-Markov Processes

342/05/95 A Sascha Hilgenfeldt, Robert Balder, Christoph Zenger: Sparse Grids: Applications to Multi-dimensional Schrödinger Problems

342/06/95 A Maximilian Fuchs: Formal Design of a Model-N Counter

342/07/95 A Hans-Joachim Bungartz, Stefan Schulte: Coupled Problems in Microsystem Technology

342/08/95 A Alexander Pfaffinger: Parallel Communication on Workstation Networks with Complex Topologies

342/09/95 A Ketil Stølen: Assumption/Commitment Rules for Data-flow Networks - with an Emphasis on Completeness

342/10/95 A Ketil Stølen, Max Fuchs: A Formal Method for Hardware/Software Co-Design

342/11/95 A Thomas Schnekenburger: The ALDY Load Distribution System

342/12/95 A Javier Esparza, Stefan Römer, Walter Vogler: An Improvement of McMillan's Unfolding Algorithm

342/13/95 A Stephan Melzer, Javier Esparza: Checking System Properties via Integer Programming

342/14/95 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Point-to-Point Dataflow Networks

342/15/95 A Andrei Kovalyov, Javier Esparza: A Polynomial Algorithm to Compute the Concurrency Relation of Free-Choice Signal Transition Graphs

342/16/95 A Bernhard Schätz, Katharina Spies: Formale Syntax zur logischen Kernsprache der Focus-Entwicklungsmethodik

342/17/95 A Georg Stellner: Using CoCheck on a Network of Workstations

342/18/95 A Arndt Bode, Thomas Ludwig, Vaidy Sunderam, Roland Wismüller: Workshop on PVM, MPI, Tools and Applications

342/19/95 A Thomas Schnekenburger: Integration of Load Distribution into ParMod-C

342/20/95 A Ketil Stølen: Refinement Principles Supporting the Transition from Asynchronous to Synchronous Communication

342/21/95 A Andreas Listl, Giannis Bozas: Performance Gains Using Subpages for Cache Coherency Control

342/22/95 A Volker Heun, Ernst W. Mayr: Embedding Graphs with Bounded Treewidth into Optimal Hypercubes

342/23/95 A Petr Jančar, Javier Esparza: Deciding Finiteness of Petri Nets up to Bisimulation

342/24/95 A M. Jung, U. Rüde: Implicit Extrapolation Methods for Variable Coefficient Problems

342/01/96 A Michael Griebel, Tilman Neunhoeffer, Hans Regler: Algebraic Multigrid Methods for the Solution of the Navier-Stokes Equations in Complicated Geometries

Reihe A

342/02/96 A Thomas Grauschopf, Michael Griebel, Hans Regler: Additive Multilevel-Preconditioners based on Bilinear Interpolation, Matrix Dependent Geometric Coarsening and Algebraic-Multigrid Coarsening for Second Order Elliptic PDEs

342/03/96 A Volker Heun, Ernst W. Mayr: Optimal Dynamic Edge-Disjoint Embeddings of Complete Binary Trees into Hypercubes

342/04/96 A Thomas Huckle: Efficient Computation of Sparse Approximate Inverses

342/05/96 A Thomas Ludwig, Roland Wismüller, Vaidy Sunderam, Arndt Bode: OMIS — On-line Monitoring Interface Specification

342/06/96 A Ekkart Kindler: A Compositional Partial Order Semantics for Petri Net Components

342/07/96 A Richard Mayr: Some Results on Basic Parallel Processes

342/08/96 A Ralph Radermacher, Frank Weimer: INSEL Syntax-Bericht

342/09/96 A P.P. Spies, C. Eckert, M. Lange, D. Marek, R. Radermacher, F. Weimer, H.-M. Windisch: Sprachkonzepte zur Konstruktion verteilter Systeme

342/10/96 A Stefan Lamberts, Thomas Ludwig, Christian Röder, Arndt Bode: PFSLib – A File System for Parallel Programming Environments

342/11/96 A Manfred Broy, Gheorghe Ştefănescu: The Algebra of Stream Processing Functions

342/12/96 A Javier Esparza: Reachability in Live and Safe Free-Choice Petri Nets is NP-complete

342/13/96 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Many-to-Many Data-flow Networks

342/14/96 A Giannis Bozas, Michael Jaedicke, Andreas Listl, Bernhard Mitschang, Angelika Reiser, Stephan Zimmermann: On Transforming a Sequential SQL-DBMS into a Parallel One: First Results and Experiences of the MIDAS Project

342/15/96 A Richard Mayr: A Tableau System for Model Checking Petri Nets with a Fragment of the Linear Time μ -Calculus

342/16/96 A Ursula Hinkel, Katharina Spies: Anleitung zur Spezifikation von mobilen, dynamischen Focus-Netzen

342/17/96 A Richard Mayr: Model Checking PA-Processes

342/18/96 A Michaela Huhn, Peter Niebert, Frank Wallner: Put your Model Checker on Diet: Verification on Local States

342/01/97 A Tobias M"uller, Stefan Lamberts, Ursula Maier, Georg Stellner: Evaluierung der Leistungsf"ahigkeit eines ATM-Netzes mit parallelen Programmierbibliotheken

342/02/97 A Hans-Joachim Bungartz and Thomas Dornseifer: Sparse Grids: Recent Developments for Elliptic Partial Differential Equations

342/03/97 A Bernhard Mitschang: Technologie f"ur Parallele Datenbanken - Bericht zum Workshop

Reihe A

342/04/97 A nicht erschienen

342/05/97 A Hans-Joachim Bungartz, Ralf Ebner, Stefan Schulte: Hierarchische Basen zur effizienten Kopplung substrukturierter Probleme der Strukturmechanik

342/06/97 A Hans-Joachim Bungartz, Anton Frank, Florian Meier, Tilman Neunhoeffer, Stefan Schulte: Fluid Structure Interaction: 3D Numerical Simulation and Visualization of a Micropump

342/07/97 A Javier Esparza, Stephan Melzer: Model Checking LTL using Constraint Programming

342/08/97 A Niels Reimer: Untersuchung von Strategien für verteiltes Last- und Ressourcenmanagement

342/09/97 A Markus Pizka: Design and Implementation of the GNU INSEL-Compiler gic

342/10/97 A Manfred Broy, Franz Regensburger, Bernhard Schätz, Katharina Spies: The Steamboiler Specification - A Case Study in Focus

342/11/97 A Christine Röckl: How to Make Substitution Preserve Strong Bisimilarity

342/12/97 A Christian B. Czech: Architektur und Konzept des Dycos-Kerns

342/13/97 A Jan Philipps, Alexander Schmidt: Traffic Flow by Data Flow

342/14/97 A Norbert Fröhlich, Rolf Schlagenhaft, Josef Fleischmann: Partitioning VLSI-Circuits for Parallel Simulation on Transistor Level

342/15/97 A Frank Weimer: DaViT: Ein System zur interaktiven Ausführung und zur Visualisierung von INSEL-Programmen

342/16/97 A Niels Reimer, Jürgen Rudolph, Katharina Spies: Von FOCUS nach INSEL - Eine Aufzugssteuerung

342/17/97 A Radu Grosu, Ketil Stølen, Manfred Broy: A Denotational Model for Mobile Point-to-Point Data-flow Networks with Channel Sharing

SFB 342 : Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

Reihe B

342/1/90 B Wolfgang Reisig: Petri Nets and Algebraic Specifications

342/2/90 B Jörg Desel: On Abstraction of Nets

342/3/90 B Jörg Desel: Reduction and Design of Well-behaved Free-choice Systems

342/4/90 B Franz Abstreiter, Michael Friedrich, Hans-Jürgen Plewan: Das Werkzeug runtime zur Beobachtung verteilter und paralleler Programme

342/1/91 B Barbara Paech1: Concurrency as a Modality

342/2/91 B Birgit Kandler, Markus Pawlowski: SAM: Eine Sortier- Toolbox -Anwenderbeschreibung

342/3/91 B Erwin Loibl, Hans Obermaier, Markus Pawlowski: 2. Workshop über Parallelisierung von Datenbanksystemen

342/4/91 B Werner Pohlmann: A Limitation of Distributed Simulation Methods

342/5/91 B Dominik Gomm, Ekkart Kindler: A Weakly Coherent Virtually Shared Memory Scheme: Formal Specification and Analysis

342/6/91 B Dominik Gomm, Ekkart Kindler: Causality Based Specification and Correctness Proof of a Virtually Shared Memory Scheme

342/7/91 B W. Reisig: Concurrent Temporal Logic

342/1/92 B Malte Grosse, Christian B. Suttner: A Parallel Algorithm for Set-of-Support
Christian B. Suttner: Parallel Computation of Multiple Sets-of-Support

342/2/92 B Arndt Bode, Hartmut Wedekind: Parallelrechner: Theorie, Hardware, Software, Anwendungen

342/1/93 B Max Fuchs: Funktionale Spezifikation einer Geschwindigkeitsregelung

342/2/93 B Ekkart Kindler: Sicherheits- und Lebendigkeitseigenschaften: Ein Literaturüberblick

342/1/94 B Andreas Listl; Thomas Schnekenburger; Michael Friedrich: Zum Entwurf eines Prototypen für MIDAS